Remarks

Status of application

Claims 1-99 were examined and stand rejected in view of prior art. The claims have been amended to further clarify Applicant's invention. Reexamination and reconsideration are respectfully requested.

The invention

A database system providing SQL extensions for automated encryption and decryption of column data is described. In one embodiment, for example, in a database system, a method of the present invention is described for providing automated encryption support for column data, the method comprises steps of: defining Structured Query Language (SQL) extensions for creating and managing column encryption keys, and for creating and managing database tables with encrypted column data; receiving an SQL statement specifying creation of a named encryption key, the named encryption key capable of encrypting multiple columns; receiving at least one SQL statement specifying creation of a database table having encrypted column data, each such SQL statement specifying a database table having particular column data encrypted with the named encryption key; and in response to a subsequent database operation that requires particular column data that has been encrypted with the named encryption key, automatically decrypting the particular column data with the named encryption key, so that the particular column data is available in decrypted form for use by the database operation.

Section 101 Rejection

Claims 37-70 stand rejected under 35 U.S.C. 101 on the basis that the claimed invention is directed to non-statutory subject matter. Here, the Examiner states that the claims are directed to a system claim but lacks any tangible hardware (i.e., that the claims are directed to software per se and are non-statutory). The claims have been amended to overcome the rejection.

Prior art rejections

A. Section 35 U.S.C. 103: Newman and Lei

Claims 1-99 are rejected under 35 U.S.C. 103(a) as being anticipated by Newman et al. (US Patent 7,266,699 82) in view of Lei et al. (US Publication 2004/0255133 A1). The Examiner's rejection of claim 1 is representative:

> As per claim 1, Newman teaches "In a database system, a method for providing automated encryption support for column data," (see Abstract and column 1 lines 46-62)
>
> "the method comprising: defining Structured Query language (SQl) extensions for creating and managing column encryption keys, and for creating and managing database tables with encrypted column data;" (column 2 lines 20-27, column 4 lines 28- 44, column 4 line 57 - column 5 line 12, column 5 lines 46-54, wherein a key management system which utilizes SQl as the standard query language provides encryption key management)
>
> "receiving an SQL statement specifying creation of a particular column encryption key;" (column 2 lines 28-40, column 7 lines 45-52, wherein a command to encrypt a column causes a key to be created)
>
> "and in response to a subsequent database operation that requires the particular column data that has been encrypted, automatically decrypting the particular column data for use by the database operation." (column 5 lines 37-54, column 8 lines 30-63, wherein encrypted data is automatically decrypted in response to an authorized user accessing the encrypted column data)
>
> While Newman teaches that encrypted database tables are able to be viewed and processed by authorized users (column 2 lines 41-57, column 7 lines 16-25, column 7 lines 45-56), Newman does not specifically teach "receiving a SQL statement
>
> specifying creation of a database table having particular column data encrypted with said particular column encryption key;"

Lei teaches "receiving an Sal statement specifying creation of a database table having particular column data encrypted with said particular column encryption key;" (paragraphs 0019, 0020, 0066, 0067 wherein a column is selected to be encrypted and stored in the database, to create encrypted data tables based on keys).

The Examiner contends that it would have been obvious for one of ordinary skill in the art to combine Newman's method of providing a transparent encryption infrastructure for databases with Lei's method of storing and updating encrypted tables. The Examiner continues: "This gives the user the ability to save encrypted data in the database. The motivation for doing so would be to more efficiently provide transparent access to user applications accessing sensitive data protected by encryption (paragraphs 0007, 0008)." For the reasons discussed below, Applicant's claimed invention may be distinguished on a variety of grounds.

Given the ever increasing security risks faced by database users, database vendors have responded with various encryption solutions, leading to a progression of improvements over time. The earliest approach was simply adding file-based encryption to database tables or entire databases. That approach is inefficient as computing resources are wasted encrypting and decrypting portions of the data that are not sensitive. Later approaches, such as described by Newman, provide improved granularity of encryption, such as encryption of individual columns. However, implementations such as Newman's toolkit represent an "add-on" or non- native approach with significant limitations. Due to a lack of integration with the underlying database system, users (e.g., database administrators) of those systems are required to assume a lot of responsibility for integrating the encryption add-on with the operations of the underlying database. Lei represents an improvement over that approach, where the underlying database system includes some level of built-in support for encryption-based database processing. Newman and Lei show that significant improvements have been made in the area of database security, since the days of simple file-based encryption. To be sure, Applicant's claimed invention pertains to the technical field of database encryption and will, of necessity, share certain features in common with other such systems. At the same time, however, it is important to understand that Applicant's claimed invention represents significant and important improvements over prior art approaches, including those of Newman and Lei, as will now be described.

The particular failing of prior art systems, such as Newman and Lei (and which is addressed by the present invention), is in the creation and management of column encryption keys. Those systems have given a lot of thought and effort to integrating encrypting and decrypting operations, but have given practically no thought to the management of the encryption keys themselves. Instead, those systems treat encryption keys as more or less an afterthought and assume that they can be obtained from some external source (e.g., a module that is not well integrated with the database engine). Significantly, those prior art systems lose an important opportunity to improve the efficiency and operation of database encryption.

In accordance with Applicant's invention, the database system has the ability to create and use a **named** encryption key which, in turn, may be used to encrypt multiple columns (i.e., with the same key), whether those columns are in the same table or in different tables. Consider the following teaching from Applicant's Specification (at [0124 - 0125]:

> A symmetric key is created using a CREATE ENCRYPTION KEY command which, in accordance with the present invention, may be implemented with the following syntax:
>
> ```
> 1: CREATE ENCRYPTION KEY keyname
> 2:    [AS DEFAULT] [FOR algorithm]
> 3:    [WITH [KEYLENGTH keysize]
> 4:       [PASSWD passphrase]
> 5:       [INIT_VECTOR [RANDOM | NULL]]
> 6:       [PAD [RANDOM | NULL]]
> ```
>
> An asymmetric key is created using a CREATE ENCRYPTION KEYPAIR command which, in accordance with the present invention, may be implemented with the following syntax:
>
> ```
> 1: CREATE ENCRYPTION KEYPAIR keypairname
> ```

```
[FOR algorithm]
[WITH [KEYLENGTH keysize]
   [PASSWD passphrase | LOGIN_PASSWD]
```

(Emphasis added.)

Here, keys are named and are unique in the namespace for the user, as emphasized at paragraph [0127] of Applicant's Specification:

> All the information related to the keys and encryption is encapsulated inside the CREATE ENCRYPTION KEY statement. As shown, the CREATE ENCRYPTION KEY statement captures properties of the column encryption key (CEK) as well as encryption properties to be used by the encryption algorithm itself. **Key properties include a keyname** and keylength, and optionally a password (PASSWD) for protecting the key. **The keyname must be unique in the user's table/view/procedure namespace in the current database**. [...]

(Emphasis added.)

Applicant's named key approach has important practical implications, as will now be described. Once a named encryption key has been created (using SQL extensions of the present invention), the named encryption key may be referenced (i.e., used) in other SQL statements. For example, other SQL statements that are creating tables may reference a particular named key as the encryption key used for column encryption of various columns that are to be encrypted. In typical use, a named key is created with a user-provided (i.e., DBA-provided) name (text string). However, as an additional feature of the present invention, naming of the key with a user-provided name is optional. If the user elects to not provide any name, the system automatically assigns a *default key* as the named key. Applicant's named encryption key approach has major advantages. Keys can be stored in a separate database from the data and can be referenced with fully qualified names. This leads to the secure storage of keys. Additionally, database performance is markedly improved for processing database queries containing inequality *sargs* (i.e., searchable arguments) on the encrypted columns and for processing database queries containing a join operation on an encrypted column between two tables (i.e., the join columns are encrypted by the same key).

In Applicant's system key management, which includes both generation and secure storage, is handled automatically (i.e., without further user intervention) by the database system, preserving security of keys and data through permissions. In other systems, the user is left to figure out key management and come up with a solution for safeguarding keys. For example, in both the Lei and Newman systems the user must come up with key management resources, and thus must turn to additional third-party (i.e., non-integrated) software for key generation and management. In Applicant's system, in contrast, all of this work is handled automatically using the built-in SQL extension to create an encryption key. Users need SELECT permission on key to use the key to encrypt a column. This can be granted by the System Security Officer (SSO). Users need DECRYPT permission on the data before they can get cleartext. This can be granted by the table owner or SSO. The above two permissions enhance the security of encrypted data. Since this process of key generation and management is kept within the confines of the database (i.e., no third party software is required), Applicant's approach is not only much more convenient but is also far more secure.

Database performance is markedly improved for processing database queries containing inequality/equality sargs (i.e., searchable arguments) on the encrypted columns and for processing database queries containing a join operation on an encrypted column between two tables (i.e., the join columns are encrypted by the same key). Ciphertext joining can be done for equality and non-equality searches if two columns are encrypted with the same key without initialization vector or random padding. This increases performance as decryption operation is not needed.

If an init vector is not used, a column can support an index and can match without decryption, for example as described in Applicant's Specification at paragraph [0132]:

Encryption properties (i.e., attributes used by the encryption algorithm) include an initialization vector (INIT_VECTOR) and padding (PAD). The init_vector NULL setting instructs the database system to omit the use of an initialization vector when encrypting. This makes the column suitable for supporting an index. When an initialization vector is used by the encryption algorithm, the ciphertext of two identical pieces of plaintext will be different, which prevents a cryptanalyst from detecting patterns of data but also renders the data on disk useless for indexing or matching without decryption. The default is to use an initialization vector, that is, init_vector random. Use of an initialization vector implies using a cipher block chaining (CBC) mode of encryption; setting init_vector NULL implies the electronic code book (ECB) mode.

If no initialization vector or random padding and columns are encrypted with the same key, it is more efficient. If no init vector and no random padding is employed, the SARG can be encrypted increasing the efficiency of search. Indexes can be created on encrypted column if no random padding or init vector. See, e.g., Applicant's Specification at [0208-0209]:

> In the currently preferred embodiment of the present invention, encrypted columns are subject to some of the restrictions described below. Referential integrity can be efficiently supported provided the columns are encrypted with no initialization vector (init_vector null) and no random padding (pad null) and the key is shared between the primary and foreign columns. [...]
>
> Encrypted columns will typically not make efficient SARGs (search arguments) because each row will need to be decrypted to match the SARG. If the column encryption is specified to omit the initialization vector (init_vector null) and no random padding (pad null) is used, the SARG can be encrypted once, increasing the efficiency of the search. Indexes may be created on encrypted columns if the encryption is specified at the column level with no initialization vector or random

padding. A create index command on an encrypted column with an initialization vector will result in an error. Indexes on encrypted columns are generally useful only for equality/non-equality matches and not for range searches. [...]

Although the above highlights the purpose and advantages of Applicant's named key approach, there is obviously no intent to write each and every benefit or advantage into Applicant's claims, as such would yield unduly narrow claims (which are not required by the cited art). In particular, Applicant's claimed invention is drawn to the named encryption key approach outlined above. Such an approach is not shown in any of the prior art of record. Nonetheless in order to expedite prosecution of the present application, the claims have been amended to further clarify such distinguishing features of Applicant's invention. For example, claim 1 now includes the claim limitations of (shown in amended form):

> receiving an SQL statement specifying creation of a ~~particular column~~ named encryption key, said named encryption key capable of encrypting multiple columns;
>
> receiving ~~an SQL statement~~ at least one SQL statement specifying creation of a database table having encrypted column data, each such SQL statement specifying a database table having particular column data encrypted with said ~~particular column~~ named encryption key; and
>
> in response to a subsequent database operation that requires ~~the~~ particular column data that has been encrypted with said named encryption key, automatically decrypting the particular column data with said named encryption key, so that the particular column data is available in decrypted form for use by the database operation.

As shown, the amended claim makes the named encryption key feature explicit in the claim. (Applicant's other independent claims have been amended in a like manner.) The benefits of this named encryption key approach are highlighted in the dependent claims including, for example, the ability to share a single encryption key among multiple columns (claim 31) and create a DEFAULT encryption key (claims 7, 36).

All told, Applicant's named encryption key approach represents a patentable advance in the area of database encryption and security. It is respectfully submitted that such an approach, as set forth in Applicant's claims, is not shown by the cited prior art. In view of the amendments to the claims (as well as clarifying remarks made above), it is believed that any rejection under Section 103 is overcome.

Any dependent claims not explicitly discussed are believed to be allowable by virtue of dependency from Applicant's independent claims, as discussed in detail above.

<u>Conclusion</u>

In view of the foregoing remarks and the amendment to the claims, it is believed that all claims are now in condition for allowance. Hence, it is respectfully requested that the application be passed to issue at an early date.

If for any reason the Examiner feels that a telephone conference would in any way expedite prosecution of the subject application, the Examiner is invited to telephone the undersigned at 408 884 1507.

Respectfully submitted,

Date: September 15, 2008 /John A. Smart/

John A. Smart; Reg. No. 34,929
Attorney of Record

408 884 1507
815 572 8299 FAX